Table of Contents

Project Description

This is a feasibility study to measure the mapping between English descriptions and low-level event traces. The research objective is to demonstrate how to automatically generate test cases from a scenario description. One purpose of this research may be to support bug triage: Given a new bug report (which includes a section for the bug reporter to delineate steps to reproduce the bug) the tool would automatically process those steps into something like a RERAN event trace that can be used to automatically confirm the bug as well as augment a regression suite.

Each tester will use a Nexus 7 tablet and a specific set of apps to generate data. There are three things to do before picking up a device:

1. Complete the steps in *Setup Environment*.
2. Prepare test scripts (i.e., scenario descriptions) using a Nexus 7 emulator running Android 4.4.2.[1]
3. Schedule time with one of the three devices using the Doodle links on Piazza.

The purpose of these steps is to maximize utilization: There are a limited number of devices and modest time windows for each tester to work with a device. Additionally, this “homework” will help familiarize you with the app so your traces on the device are more “natural” rather than introducing unnatural delays between events to record your actions. Use the instructions in *Guidance on Scenario Descriptions* to build a well-formed corpus. After gaining some familiarity with the app(s) on an emulator (or your personal device) and building scenario descriptions, you should pick-up the device from the SEMERU lab at your scheduled time.

When you receive the device, you may have to connect it to a wireless access point depending on your location. Otherwise, the device is ready to record your traces! <u>Place the device on a flat surface in portrait orientation.</u> Select one of the following five applications: Calculator, Frex, GM Dice, Mirakel or Wikipedia. Then follow the instructions in *Record Traces* to capture low-level event traces. <u>Note that you must clear all cached app data before starting a new scenario.</u> There are also instructions in this section for validating your description against the trace. (There should be one sentence in the description for each user action in the trace.) If the description does not match the trace, then you can replay your trace and observe the recorded user actions to spot discrepancies with your sentences.

[1] If you have not worked with an emulator, there is guidance on configuring a new emulator to build scenario descriptions in an appendix. Alternatively, you may install the app(s) on your personal Android device.

If you need to replay your trace, then follow the instructions in *Replay Traces*. Before you replay your first event trace, you must `chown` a directory on the device so you can push files from your host to the device. This only needs to be done once for a host. (The details are in *Replay Traces*.)

Please consider the following requirements:

- The text editor you use to transcribe your scenario descriptions must use ASCII text.
- Scenario descriptions shall follow the *Guidance on Scenario Descriptions*.
- Scenario description files shall be named `description.txt`.
- Run each scenario description AT LEAST three times to generate AT LEAST three distinct event traces. So, each one of your scenarios will have a minimum of four files (one description and at least three event traces).
- Event traces shall be named `record1.log`, `record2.log`, `record3.log`, etc. depending on the number of the trace.
- Organize your files using `<last-name>/<app-name>/<scenario-name>/`. In other words, you should have one root directory (with your last name in lowercase letters [a-z]) which contains five subdirectories where each subdirectory is named after an app (either “calculator,” “frex,” “gm-dice,” “mirakel” or “wikipedia”). Each app directory contains several subdirectories where each subdirectory is named after a scenario. The scenario name should summarize feature(s) tested in the description, and you are limited to lowercase letters [a-z], digits [0-9], hyphens and periods. (Do NOT use spaces; use hyphens to delimit lexemes.) For example,

```
$ ls
white
$ cd white/
$ ls
calculator frex gm-dice mirakel wikipedia
$ ls calculator/
add-two-numbers subtract-two-numbers multiply-two-numbers divide-two-numbers . . .
$ cd add-two-numbers/
$ ls
description.txt record1.log record2.log record3.log . . .
$
```

- Each scenario shall test one or more features. Generate data for as many scenarios as you can in the time allotted,[2] but choose your scenarios such that each successive scenario tests functionality that is “orthogonal” to your previous scenarios. (Think Gram-Schmidt!) The goal is to obtain a comprehensive set of traces that test many different features.
- Create a zipped archive of your data by running

```
$ tar czf <last-name>.tgz <last-name>/
```

- Email the compressed tarball to mlinarev@cs.wm.edu with the subject line “RERAN Data”.

Maintain your data and periodically email a compressed tarball of your working directory `<last-name>`. We will run `$ cp -ru` to update our copy of your data when you send us snapshots of your archive.

[2] It takes approximately 10 minutes to generate a scenario description and trace that description.

Setup Environment (for Ubuntu 13.10 or Mac OS X)

1. Install the required packages by running

```
$ cd
$ sudo apt-get install openjdk-7-jdk
$ sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0 lib32stdc++6 libgl1-mesa-dev
```

2. Download the ADT Bundle for the Linux 64-bit platform from Android Developers to `$HOME/Downloads`
3. Install the Android SDK by running[3]

```
$ unzip Downloads/adt-bundle-linux-x86_64-20131030.zip –d Development
$ mv Development/adt-bundle-linux-x86_64-20131030 Development/adt
$ export PATH=$PATH:$HOME/Development/adt/sdk/platform-tools:$HOME/Development/adt/sdk/tools
```

4. You should add the export command to your `.profile`
5. Download a compressed tarball and extract the archive, which contains three directories, by running[4]

```
$ wget -P Downloads/ www.cs.wm.edu/~pj/reran-files.tgz
$ tar xzf Downloads/reran-files.tgz -C Development/
```

[3] The filename of the zip file in the following instructions may be different depending on the date that you download the SDK.

[4] If you are running a terminal emulator in Mac OS X, use `$ curl www.cs.wm.edu/~pj/reran-files.tgz -o ~/Downloads/reran-files.tgz` instead of `wget`.

Guidance on Scenario Descriptions

You must capture descriptive text that specifies each and every one of your actions on the app from the time you start recording until the time you stop recording. **This description must be comprised of well-formed English sentences, one sentence per line, ending with a period.** Each sentence should contain a subject, verb and component as well as any number of adjectives, adverbs, prepositional phrases and participial phrases describing the verb or component.[5] Specifically:

Subject: Use the first-person singular pronoun “I” as the subject.

Verb: The verb is your action, and you are limited to `touched`, `longtouched`, `swiped` and `typed`. `swiped` should be immediately followed by either `up`, `down`, `left` or `right`. Do NOT use multi-touch gestures such as `zoomed` and `pinched`. `typed` should be followed by a string of characters enclosed in double quotes. You are limited to the following character set: lowercase letters [a-z], digits [0-9] and spaces. `typed` is essentially a macro that expands one sentence into a sequence of sentences where the number of sentences is equal to the length of the string; however, the sentence that precedes the `typed` action must describe how you brought focus to the component. For example, the sequence of sentences on the left (from the tester) will be parsed into the sequence of sentences on the right:

I touched the Search text field.
I typed “new york” in the Search text field.
I touched the “New York” list item.

I touched the Search text field.
I touched the “n” button on the keyboard.
I touched the “e” button on the keyboard.
I touched the “w” button on the keyboard.
I touched the <Space> button on the keyboard.
I touched the “y” button on the keyboard.
I touched the “o” button on the keyboard.
I touched the “r” button on the keyboard.
I touched the “k” button on the keyboard.
I touched the “New York” list item.

[5] While every sentence must contain a subject and a verb, there are some rare cases where a sentence may not contain a component. For example, neither “I swiped up on the touchscreen.” nor “I touched a hyperlink called ‘Beltran Fights Off Slumps With a Bat at His Bedside’.” contain a component.

Note that the first sentence is necessary since it brings focus to the text field. Also, you MUST type your complete query. So, if relevant results appear after typing “new y” then finish your query before selecting the “New York” list item. Moreover, if you mess up while typing your string, then you must use the backspace to clear the mistake; do NOT touch the cursor into the proper position. For example,

> I touched the Search text field.
> I typed “new yptl” in the Search text field.
> I touched the Backspace button on the keyboard.
> I touched the Backspace button on the keyboard.
> I touched the Backspace button on the keyboard.
> I typed “ork” in the Search text field.
> I touched the “New York” list item.

Component: You are limited to the following components: `list item`, `grid item`, `button`, `text field`, `checkbox`, `radio button`, `toggle button`, `spinner`, `picker`, `seek bar` and `image`.[6]

Descriptors: The adjectives, adverbs, prepositional phrases and participial phrases should specify the verb or component. For instance, describe the location on the touchscreen where the action was performed, the color of the component, any text on the component, etc.

There should be a bijection from the sentences in your description to the user actions captured in the RERAN log. (Remember, the `typed` macro is a convenience. It saves the tester from having to include a sentence for each character in a string.) And the sentences in your description must be in the same order as the user actions in the trace. Here is an example of a scenario description:

> I touched the button with the magnifying glass at the top of the touchscreen.
> I touched the Search articles text field.
> I typed “new york yankees” in the Search articles text field.
> I touched a hyperlink called “Beltran Fights Off Slumps With a Bat at His Bedside”.
> I swiped up on the touchscreen.

[6] Each component in this list is a valid View object. (See the User Interface Guide for images of each component.) If you discover a component that you engage during testing that is not in this list, then verify it is a valid View object by referring to the User Interface Guide. If it is a valid component, then include it in your description and send an email to mgwhite@cs.wm.edu.

I swiped up on the touchscreen.
I swiped up on the touchscreen.
I swiped up on the touchscreen.
I swiped up on the touchscreen.
I swiped up on the touchscreen.
I swiped up on the touchscreen.
I swiped up on the touchscreen.
I touched the Back button.
I swiped up on the touchscreen.

RERAN is an incredibly precise logger that captures the number of microseconds between events. Try to script your scenario ahead of time (on an emulator) so the timing in your trace is “natural” instead of unnaturally taking breaks to log your descriptions. This may require gaining some familiarity with the AUT ahead of time to discover features that you would like to test. Please verify the number of sentences for each scenario matches the number of user actions in the corresponding RERAN log by replaying the script and observing the actions that were recorded and comparing them to your description. `count_actions.py` is a script we provide to help test this postcondition.

Record Traces

1. Verify the device is plugged into a USB port (or launch the emulator) and place it on a flat surface in portrait orientation
2. Clear all cached app data before recording a new getevent trace via Settings > Storage > Cached data > OK
3. Open the AUT and leave it open on its main activity to start a new scenario
4. Start recording by opening a new terminal and running (where `<n>` represents the number of the trace, e.g. `record1.log`)[7]

```
$ adb shell getevent -t > ~/Documents/record<n>.log
```

5. Do stuff. And keep track of what you do using *Guidance on Scenario Descriptions*
6. When you're done, stop recording by pressing Ctrl+C in the terminal running getevent
7. Validate your description against your trace by running

```
$ cd ~/Development/scripts/
$ ./count_actions.py ~/Documents/<scenario-name>.txt ~/Documents/record<n>.log
I touched the Search text field................................................... touch action: duration=0.071167, num_coords=2
I typed "android" in the Search text field.
    a............................................................................. touch action: duration=0.064697, num_coords=1
    n............................................................................. touch action: duration=0.051941, num_coords=1
    d............................................................................. touch action: duration=0.088867, num_coords=1
    r............................................................................. touch action: duration=0.063874, num_coords=1
    o............................................................................. touch action: duration=0.048156, num_coords=1
    i............................................................................. touch action: duration=0.053894, num_coords=1
    d............................................................................. touch action: duration=0.087524, num_coords=1
I touched the Go button on the keyboard........................................... touch action: duration=0.036164, num_coords=1
I touched the "Android (operating system)" list item.............................. touch action: duration=0.053192, num_coords=1
I swiped up, to scroll down on the page........................................... touch action: duration=0.658478, num_coords=84
I touched the "Multi-touch" hyperlink for the default user interface.............. touch action: duration=0.090790, num_coords=5
I swiped up, to scroll down on the page........................................... touch action: duration=0.405640, num_coords=52
I touched the "Implementations" hyperlink......................................... touch action: duration=0.082641, num_coords=3
I swiped up, to scroll down on the page........................................... touch action: duration=1.207489, num_coords=164
I swiped up, to scroll down on the page........................................... touch action: duration=0.767487, num_coords=100
I swiped up, to scroll down on the page........................................... touch action: duration=0.709167, num_coords=93
I swiped up, to scroll down on the page........................................... touch action: duration=0.530456, num_coords=59
I swiped up, to scroll down on the page........................................... touch action: duration=0.379150, num_coords=44
I swiped up, to scroll down on the page........................................... touch action: duration=0.429870, num_coords=56

Success: It looks like they match up!
```

8. If the parallel text pass, then the scenario description and the trace are ready to be submitted. Otherwise, you should replay

[7] Be careful with overwriting previous traces.

your RERAN trace and observe any discrepancies with your description. Note that the values for duration and the number of coordinates with the corresponding user action in the description may help localize discrepancies

Replay Traces

If you are replaying a trace using an adb process running on a new host, then you must begin with Step 1. If you already changed ownership of `/data/local` on the device using an adb process running on your host, then begin with Step 4 to replay your trace.

1. Verify the device is rooted by running

```
$ adb shell su -c ls -l /data/local
```

2. Click the Grant button on the Superuser request message
3. Give the adb shell write access to `/data/local` by running[8]

```
$ adb shell su -c chown shell:shell /data/local
```

4. Translate your getevent traces by running[9] (where `<n>` represents the number of the trace, e.g. `record1.log` and `replay1.log`)

```
$ cd ~/Development/reran/
$ java Translate ~/Documents/record<n>.log ~/Documents/replay<n>.log
```

5. Push the translated trace to the device/emulator by running

```
$ adb push ~/Documents/replay<n>.log /data/local
```

6. Open the AUT and leave it open on its main activity
7. Run your scenario using RERAN on the device/emulator by running

```
$ adb shell /data/local/replay /data/local/replay<n>.log
```

[8] Changing ownership of `/data/local` on the device from `root:root` to `shell:shell` will allow `adb` to push files to `/data/local`. (This does not have to be done for an emulator.)

[9] If it complains about asterisks, then delete the first few lines from `record<n>.log` that start with asterisks and try again. This is output from the adb server; it's not from `getevent`.

Appendix A: Setup, Launch and Configure an Emulator

Setup Emulator

1. Download Android 4.4.2 using the Android SDK Manager by running

```
$ android sdk
```

2. Check the box for Android 4.4.2 (API 19)
3. Click the Install <n> packages... button
4. Select the Accept License radio button
5. Click the Install button
6. After the Android SDK Manager is done loading packages, click the Close button to close the log window
7. Close the Android SDK Manager
8. Create the emulator using the Android Virtual Device Manager by running

```
$ android avd
```

9. In the Android Virtual Devices tab, click the New… button to create a new AVD
10. Create a new Android Virtual Device (AVD) with the following information

```
AVD Name: jerry
Device: Nexus 7 (7.02”, 1200 x 1920: xhdpi)
Target: Android 4.4.2 - API Level 19
CPU/ABI: ARM (armeabi-v7a)
Keyboard: Hardware keyboard present
Skin: No skin
Front Camera: None
Back Camera: None
Memory Options: RAM: 2048 VM Heap: 64
Internal Storage: 200 MiB
SD Card:
Emulation Options: Use Host GPU
```

11. Click the OK button
12. A window will display the result of creating AVD jerry. Click the OK button

13. Close the Android Virtual Device Manager

Launch Emulator

1. There are two ways to launch an emulator (using the Android Virtual Device Manager GUI or the CLI). You can launch AVD jerry using the CLI by running

```
$ emulator @jerry
```

If this is the first time you are launching the emulator, then you can choose whether or not to send usage statistics to Google. (This setting can be changed in the ddms tool under File > Preferences > Usage Stats.) Click the Proceed button.

Install RERAN and the AUT on the Emulator

1. Before issuing adb commands, it is helpful to know what device/emulator instances are connected to the adb server. You can generate a list of attached devices/emulators by running

```
$ adb devices
```

2. Put the RERAN client on the emulator and install the AUT on the emulator by running[10]

```
$ adb push Development/reran/replay /data/local
$ adb install Development/apps/<application-under-test>.apk
```

You can choose to let Google regularly check installed apps for potentially harmful behavior. This setting can be changed in Settings > Security > Verify apps

[10] If there's only one emulator running or only one device connected, the adb command is sent to that device by default. If multiple emulators are running and/or multiple devices are attached, you need to use the -d, -e, or -s option to specify the target device to which the command should be directed. Ref. Android Debug Bridge